

1 **WHAT IS CLAIMED IS:**

2 1. A method of operating a file server in a data network, said method
3 comprising:

4 (a) the file server receiving a request for metadata about a file to be accessed, the
5 request being received from a data processing device in the data network; and

6 (b) in response to the request for metadata, the file server granting to the data
7 processing device a lock on at least a portion of the file, and returning to the data
8 processing device metadata of the file including information specifying data storage
9 locations in the file server for storing data of the file.

10
11 2. The method as claimed in claim 1, wherein the file server includes a data
12 storage device including the data storage locations, and a data mover computer for
13 managing locks on files having data stored in said data storage device, wherein the data
14 storage device stores metadata of a plurality of files having file data stored in the data
15 storage device, the data mover computer is coupled to the data storage device for transfer
16 of the metadata between the data storage device and the data mover computer, the data
17 mover computer has a random access memory, and the method includes the data mover
18 computer maintaining a metadata cache in the random access memory, and the method
19 includes the data mover computer accessing the metadata cache for obtaining the
20 metadata that is returned to the data processing device.

21
22 3. The method as claimed in claim 1, wherein a plurality of data processing
23 devices in the data network share read-write access to the file, and the file server grants
24 respective read locks and write locks to the data processing devices in the data network.

25
26 4. The method as claimed in claim 1, wherein the data processing device
27 writes data to the data storage locations in the file server, modifies the metadata from the
28 file server in accordance with the data storage locations in the file server to which the
29 data is written, and sends the modified metadata to the file server.

1 5. The method as claimed in claim 4, wherein the data processing device
2 sends the modified metadata to the file server after the data processing device writes the
3 data to the data storage of the file server.

4
5 6. The method as claimed in claim 1, wherein the data processing device has
6 a cache memory for caching the metadata of the file including a version identifier
7 associated with the metadata of the file, and wherein the data processing device includes
8 the version identifier in the request for access to the file, the file server compares the
9 version identifier from the data processing device to a version identifier of a most recent
10 version of the metadata of the file, and the file server returns the most recent version of
11 the metadata of the file to the data processing device when the comparison of the version
12 identifier from the data processing device to the version identifier of the most recent
13 version of the metadata of the file indicates that the metadata of the file cached in the
14 cache memory of the data processing device is not the most recent metadata of the file.

15
16 7. The method as claimed in claim 6, wherein the version identifier is a
17 number that is incremented when the metadata of the file is modified.

18
19 8. A method of operating a file server and a client in a data network, said
20 method comprising:

- 21 (a) the client sending to the file server at least one request for access to a file;
22 (b) the file server receiving said at least one request for access to the file, granting
23 to the client a lock on at least a portion of the file, and sending to the client metadata of
24 the file including information specifying data storage locations in the file server for
25 storing data of the file;
26 (c) the client receiving from the file server the metadata of the file, using the
27 metadata of the file to produce at least one data access command for accessing the data
28 storage locations in the file server, and sending the data access command to the file server
29 to access the data storage locations in the file server; and
30 (d) the file server responding to the data access command by accessing the data

1 storage locations in the file server.

2
3 9. The method as claimed in claim 8, wherein the file server includes a data
4 storage device including the data storage locations, and a data mover computer for
5 managing locks on files having data stored in said data storage device, and wherein the
6 client sends to the data mover computer said at least one request for access to the file, the
7 data mover computer responds to said at least one request for access to the file by
8 returning to the client the metadata of the file, and wherein the client sends the data
9 access command to the data storage device over a data transmission path that bypasses
10 the data mover computer.

11
12 10. The method as claimed in claim 9, wherein the data storage device stores
13 metadata of a plurality of files having file data stored in the data storage device, the data
14 mover computer is coupled to the data storage device for transfer of the metadata between
15 the data storage device and the data mover computer, the data mover computer has a
16 random access memory, and the method includes the data mover computer maintaining a
17 metadata cache in the random access memory, and the method includes the data mover
18 computer accessing the metadata cache for obtaining the metadata that is sent to the
19 client.

20
21 11. The method as claimed in claim 8, wherein a plurality of clients in the data
22 network share read-write access to the file, and the file server grants respective read locks
23 and write locks to the clients in the data network.

24
25 12. The method as claimed in claim 8, wherein the lock on at least a portion of
26 the file granted by the file server to the client is not granted to any particular application
27 process of the client, and wherein the client has a lock manager that grants a local file
28 lock to a particular application process that accesses the file.

29
30 13. The method as claimed in claim 8, wherein the client has a lock manager

1 that responds to a request from an application process of the client for access to the file
2 by granting to the application process a local file lock on at least a portion of the file, and
3 then sending to the file server said at least one request for access to the file.
4

5 14. The method as claimed in claim 8, wherein the method includes
6 dynamically linking application programs of the client with input-output related operating
7 system routines of the client, the input-output related operating system routines
8 intercepting file access calls from client application processes to send file access requests
9 to the file server to obtain from the file server locks upon at least a portion of each of the
10 files, to obtain metadata for producing data access commands for accessing data storage
11 in the file server, to produce the data access commands from the metadata, and to send
12 the data access commands to the file server in order to access the data storage of the file
13 server.
14

15 15. The method as claimed in claim 8, wherein the data access command is a
16 write command for a write operation upon at least a portion of the file, and wherein the
17 method includes the client writing the data to the data storage locations in the file server,
18 modifying the metadata from the file server in accordance with the write operation upon
19 at least a portion of the file, and sending the modified metadata to the file server.
20

21 16. The method as claimed in claim 15, wherein the client sends the modified
22 metadata to the file server after the client writes the data to the data storage of the file
23 server.
24

25 17. The method as claimed in claim 16, wherein the client performs
26 asynchronous write operations upon the data storage locations of the file server, and
27 wherein the client sends the modified metadata to the file server in response to a commit
28 request from an application process of the client.
29

30 18. The method as claimed in claim 16, wherein the client performs

1 asynchronous write operations upon the data storage locations of the file server, and
2 wherein the client sends the modified metadata to the file server when the client requests
3 the file server to close the file.
4

5 19. The method as claimed in claim 8, wherein the client has a cache memory
6 for caching the metadata of the file including a version identifier associated with the
7 metadata of the file, and wherein the client includes the version identifier in the request
8 for access to the file, the file server compares the version identifier from the client to a
9 version identifier of a most recent version of the metadata of the file, and the file server
10 returns the most recent version of the metadata of the file to the client when the
11 comparison of the version identifier from the client to the version identifier of the most
12 recent version of the metadata of the file indicates that the metadata of the file cached in
13 the cache memory of the client is not the most recent metadata of the file.
14

15 20. The method as claimed in claim 19, wherein the version identifier is a
16 number that is incremented when the metadata of the file is modified.
17

18 21. A file server comprising:
19 at least one data storage device for storing a file system; and
20 a data mover computer coupled to the data storage device for exchange of
21 metadata of files in the file system, the data mover computer having at least one network
22 port for exchange of control information and metadata of files in the file system with data
23 processing devices in the data network, the control information including metadata
24 requests;

25 wherein the data storage device has at least one network port for exchange of data
26 with the data processing devices in the data network over at least one data path that
27 bypasses the data mover computer; and

28 wherein the data mover computer is programmed for responding to each metadata
29 request for metadata of a file from each data processing device by granting to said each
30 data processing device a lock on at least a portion of the file, and returning to said each

1 data processing device metadata of the file including information specifying data storage
2 locations in the data storage device for storing data of the file.

3
4 22. The file server as claimed in claim 21, wherein the data mover computer is
5 programmed to receive modified metadata from said each data processing device, and
6 write the modified metadata to the data storage device.

7
8 23. The file server as claimed in claim 21, wherein the data mover computer
9 has a random access memory, and the data mover computer is programmed for
10 maintaining a metadata cache in the random access memory, and the data mover
11 computer is programmed for accessing the metadata cache for obtaining the metadata that
12 is returned to said each data processing device.

13
14 24. The file server as claimed in claim 23, wherein the data mover computer is
15 programmed for receiving modified metadata from said each data processing device, and
16 writing the modified metadata to the metadata cache in the random access memory.

17
18 25. The file server as claimed in claim 21, wherein the data mover computer is
19 programmed for receiving a metadata version identifier in the metadata request to a
20 version identifier of a most recent version of the metadata of the file, and for returning the
21 most recent version of the metadata of the file to said each data processing device when
22 the comparison indicates that the metadata version identifier in the metadata request fails
23 to identify the most recent version of the metadata of the file.

24
25 26. The file server as claimed in claim 25, wherein the version identifier is a
26 number, and the data mover computer is programmed to increment the version identifier
27 when the metadata of the file is modified.

28
29 27. A data processing system comprising, in combination;
30 a file server; and

1 a plurality of clients linked by a data network to the file server;
2 wherein the file server is programmed for receiving from each client at least one
3 request for access to a file, for granting to said each client a lock on at least a portion of
4 the file, and for sending to said each client metadata of the file including information
5 specifying data storage locations in the file server for storing data of the file;
6 wherein said each client is programmed for using the metadata of the file to
7 produce at least one data access command for accessing data of the file; and
8 wherein the file server is programmed for receiving from said each client said at
9 least one data access command for accessing data of the file by accessing the data storage
10 locations in the file server.

11
12 28. The data processing system as claimed in claim 27, wherein the file server
13 includes a data storage device including the data storage locations, and a data mover
14 computer programmed for managing locks on files having data stored in said data storage
15 device, wherein the data mover computer has a network port for receipt of file access
16 requests from clients, and wherein the data storage device has a network port for receipt
17 of data access commands from said clients over at least one data transmission path that
18 bypasses the data mover computer.

19
20 29. The data processing system as claimed in claim 28, wherein the data
21 storage device stores metadata of a plurality of files having file data stored in the data
22 storage device, the data mover computer is coupled to the data storage device for the
23 transfer of the metadata between the data storage device and the data mover computer, the
24 data mover computer has a random access memory, and the data mover computer is
25 programmed for maintaining a metadata cache in the random access memory, and for
26 accessing the metadata cache for obtaining the metadata that is sent to said each client.

27
28 30. The data processing system as claimed in claim 27, wherein a plurality of
29 clients in the data network share read-write access to files stored in data storage of the file
30 server, and the file server is programmed to grant respective read locks and write locks to

1 the clients in the data network.

2
3 31. The data processing system as claimed in claim 27, wherein the lock on at
4 least a portion of the file granted by the file server to the client is not granted to any
5 particular application process of said each client, and wherein said each client has a lock
6 manager for granting a local file lock to a particular application process that accesses the
7 file.

8
9 32. The data processing system as claimed in claim 27, wherein said each
10 client has a lock manager for responding to a request from an application process of said
11 each client for access to the file by granting to the application process a local file lock on
12 at least a portion of the file, and then sending to the file server said at least one request for
13 access to the file.

14
15 33. The data processing system as claimed in claim 27, wherein said client is
16 programmed with input-output related operating system routines for intercepting file
17 access calls from client application processes for sending file access requests to the file
18 server.

19
20 34. The data processing system as claimed in claim 33, wherein the data
21 access commands include at least one write command for a write operation upon at least a
22 portion of at least one file, and wherein the client is programmed for writing data to data
23 storage locations in the file server, modifying the metadata from the file server in
24 accordance with the write operation upon at least a portion of said at least one file, and
25 sending the modified metadata to the file server.

26
27 35. The data processing system as claimed in claim 27, wherein said each
28 client has a cache memory for caching the metadata of the file including a version
29 identifier associated with the metadata of the file, and wherein said each client is
30 programmed to include the version identifier in the request for access to the file, the file

1 server is programmed to compare the version identifier from said each client to a version
2 identifier of a most recent version of the metadata of the file, and the file server is
3 programmed to return the most recent version of the metadata of the file to the client
4 when the comparison of the version identifier from said each client to the version
5 identifier of the most recent version of the metadata of the file indicates that the metadata
6 of the file cached in the cache memory of said each client is not the most recent metadata
7 of the file.

8
9 36. A program storage device containing a program for a file server, the file
10 server having at least one data storage device for storing a file system, and having at least
11 one network port for exchange of control information and metadata of files in the file
12 system with at least one data processing device, the control information including
13 metadata requests, wherein the program is executable by the file server for responding to
14 each metadata request for metadata of a file by granting to said each data processing
15 device a lock on at least a portion of the file, and returning to said each data processing
16 device metadata of the file including information specifying data storage locations in the
17 data storage device for storing data of the file.

18
19 37. The program storage device as claimed in claim 36, wherein the program
20 is executable by the file server for receiving modified metadata from the data processing
21 device and writing the modified metadata to data storage of the file server.

22
23 38. The program storage device as claimed in claim 36,
24 wherein the file server has nonvolatile data storage containing the file system and
25 metadata of files in the file system, and the file server has a random access memory; and
26 wherein the program is executable by the file server for maintaining a metadata
27 cache in the random access memory, and for accessing the metadata cache for obtaining
28 the metadata that is returned to the data processing device.

29
30 39. The program storage device as claimed in claim 38, wherein the program

1 is executable by the file server for receiving modified metadata from the data processing
2 device, and writing the modified metadata to the metadata cache in the random access
3 memory.

4
5 40. The program storage device as claimed in claim 38, wherein the program
6 is executable by the file server for receiving a metadata version identifier in the metadata
7 request to a version identifier of a most recent version of the metadata of the file, and for
8 returning the most recent version of the metadata of the file to the data processing device
9 when the comparison indicates that the metadata version identifier in the metadata request
10 fails to identify the most recent version of the metadata of the file.

11
12 41. The program storage device as claimed in claim 40, wherein the version
13 identifier is a number, and the program is executable by the file server for incrementing
14 the version identifier when the metadata of the file is modified.

15
16 42. A program storage device containing a program for a data processing
17 device that is a client in a data network, the program being executable by the client to
18 enable application programs of the client to access files in data storage of at least one file
19 server in the data network, the program being executable in response to a call from an
20 application program for access to data of a file by sending to the file server a metadata
21 request for metadata of the file including information specifying data storage locations for
22 data of the file in the file server, receiving the metadata of the file from the file server,
23 using the metadata of the file to produce at least one data access command for accessing
24 the data storage locations in the file server, and sending the data access command to the
25 file server to access the data storage locations in the file server.

26
27 43. The program storage device as claimed in claim 42, wherein the program
28 is executable by the client in response to the call from the application program by
29 granting a local file lock to a particular application process that is executing the
30 application program, and then sending to the file server the metadata request for metadata

1 of the file.

2
3 44. The program storage device as claimed in claim 42, wherein the program
4 includes input-output related operating system routines for intercepting file access calls
5 from the client application programs.
6

7 45. The program storage device as claimed in claim 42, wherein the data
8 access command is a write command for a write operation upon at least a portion of the
9 file, and wherein the program is executable by the client for writing the data to the data
10 storage locations in the file server, modifying the metadata from the file server in
11 accordance with the write operation upon at least a portion of the file, and sending the
12 modified metadata to the file server.
13

14 46. The program storage device as claimed in claim 45, wherein the program
15 is executable by the client for sending the modified metadata to the file server after the
16 client writes the data to the data storage of the file server.
17

18 47. The program storage device as claimed in claim 45, wherein the program
19 is executable for sending the modified metadata to the file server in response to a commit
20 request from the application program.
21

22 48. The program storage device as claimed in claim 45, wherein the program
23 is executable for sending the modified metadata to the file server when the client requests
24 the file server to close the file.
25

26 49. The program storage device as claimed in claim 42, wherein the client has
27 a cache memory for caching the metadata of the file including a version identifier
28 associated with the metadata of the file, and wherein the program is executable by the
29 client for including the version identifier in the metadata request that is sent to the file
30 server.